Contents

Chapter 1

Introduction

> *"Jealous stepmother and sisters; magical aid by a beast; a marriage won by gifts magically provided; a bird revealing a secret; a recognition by aid of a ring; or show; or what not; a démouement of punishment; a happy marriage - all those things, which in sequence, make up Cinderella, may and do occur in an incalculable number of other combinations. "*
>
> — MR. Cox ***1893****, Cinderella: Three hundred and forty-five variants [*? *]*

The Web Consortium (W3C) standarized bytecode for the web environment with the WebAssembly (Wasm) language in 2015. Wasm allows web browsers to execute existing programs or libraries written in other languages, such as C/C++ and Rust. Beyond web environments, WebAssembly evolves to be part of Edge-Cloud computing platforms [? ?]. Despite being designed for sandboxing and secure execution, it is not exempt from vulnerabilities [?]. For example, WebAssembly engines are vulnerable to speculative execution [?], and C/C++ source code vulnerabilities might be ported to Wasm binaries [?].

One strategy to hide such vulnerabilities is to move them in time as a preemptive solution. The goal is to make potential vulnerabilities available only in a time window. This makes potential attackers not hit what they cannot see. This strategy is usually called Moving Target Defense (MTD) [? ?]. MTD for software is a collection of techniques that aim to improve the security of a system by constantly rotating its vulnerable programs from one variant to another. A program variant should be different from the original program but functionally equivalent to it. By rotating the deployment and execution between the program variants, a potential attacker needs more efforts to perform the same attack for all variants [?]. Thus, one premise for effectively implementing MTD for a given program is the need for the program variants.

In MTD, Software Diversification is the process of finding, creating, and deploying program variants. Usually, program variants could be found in the wild in a phenomenon called natural diversity [?]. In the case of WebAssembly, since it is a novel technology, there is no natural diversity. Thus, effective MTD cannot be implemented due to the lack of program variants. This work proposes to create program variants for WebAssembly artificially. Therefore, we aim to generate artificial software diversification for WebAssembly. To reach such a goal, we answer three research questions enunciated in the following.

1.1 Research questions

In this section, we present our three research questions. Our research questions are formulated by merging our publications and experiences during the creation of Software Diversification for WebAssembly.

RQ_1 **To what extent can we artifically generate program variants for WebAssembly?**
With this research question, we quantitatively assess the static differences between program variants created by our approach. We answer this question at the population level, where a program population is the collection of one original program and its generated variants. We aim to investigate the code properties that increases(or diminishes) generated diversification at population level.

RQ_2 **To what extent are the generated variants dynamically different?**
With this research question, we complement RQ_1. We aim to investigate the impact on execution traces and execution times of the generated program variants.

RQ_3 **To what extent do the artificial variants exhibit different execution times on Edge-Cloud platforms?**
With this research question, we aim to investigate the impact of Software Diversification for WebAssembly in an emerging technology, Edge-Cloud computing. We evaluate the impact of a novel multivariant execution approach on real-world WebAssembly programs in a world-wide scale experiment.

1.2 Contributions

This thesis contributes through four milestones. First, as a theoretical contribution, we summarize the code transformations used to artifically generate software diversification through an exhaustive literature review. Consequently, we highlight the lack of diversification techniques for WebAssembly. In addition, we discuss the incorporation of *constant inferring* as a new transformation. Second, as a technical contribution, we provide two tools, CROW [**?**] and MEWE [**?**]. Besides, we summarize the main challenges faced during their implementation. Third, we propose a methodology to quantitatively evaluate the impact of our tools, assessing the creation of artificial software diversification for WebAssembly. Fourth and final, we empirically demonstrate the impact of our technique by evaluating the static and dynamic behavior of the generated diversification.

1.3 Publications

This work is based on the following publications:

P_1 Superoptimization of WebAssembly Bytecode [**?**]
Javier Cabrera-Arteaga, Shrinish Donde, Jian Gu, Orestis Floros, Lucas Satabin, Benoit Baudry, Martin Monperrus
Programming 2020, MoreVMs'20

P_2 CROW: Code Diversification for WebAssembly [**?**]
Javier Cabrera-Arteaga, Orestis Floros, Oscar Vera-Pérez, Benoit Baudry, Martin Monperrus
NDSS 2021, MADWeb

P_3 Multi-Variant Execution at the Edge [**?**]
Javier Cabrera-Arteaga, Pierre Laperdrix, Martin Monperrus, Benoit Baudry
Under review

Other publications and talks

1. Scalable Comparison of JavaScript V8 Bytecode Traces [**?**]
Javier Cabrera-Arteaga, Martin Monperrus, Benoit Baudry
SPLASH 2019, VMIL

2. (Talk) Wasm-mutate: Fuzzing WebAssembly Compilers with E-Graphs
Javier Cabrera-Arteaga, Nicholas Fitzgerald, Martin Monperrus, Benoit Baudry
PLDI 2022, EGRAPHs

Thesis layout

This dissertation is organized in five chapters including this. **??** presents background and the state of the art for WebAssembly and Artificial Software Diversification. **??** describes our technical contributions, faced challenges and engineering decisions carried out to implement our artifacts. **??** describes the methodology followed to answer the three main research questions driving this thesis. **??** details the main results of this work. **??** concludes and discuss future work. In addition, this dissertation contains the collection of research papers previously mentioned in this chapter.

Bibliography

[] Stiévenart,Q., De Roover,C., and Ghafari,M. (2022). Security risks of porting c programs to webassembly. In *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, SAC '22, page 1713–1722, New York, NY, USA. Association for Computing Machinery.

[] Harrand,N. (2022). *Software Diversity for Third-Party Dependencies.* PhD thesis, KTH, Software and Computer systems, SCS. QCR 20220413.

[] Spies,B. and Mock,M. (2021). An evaluation of webassembly in non-web environments. In *2021 XLVII Latin American Computing Conference (CLEI)*, pages 1–10.

[] NSA (2021). National Cyber Leap Year.

[] Narayan,S., Disselkoen,C., Moghimi,D., Cauligi,S., Johnson,E., Gang,Z., Vahldiek-Oberwagner,A., Sahita,R., Shacham,H., Tullsen,D., et al. (2021). Swivel: Hardening webassembly against spectre. In *USENIX Security Symposium.*

[] Cabrera Arteaga,J., Laperdrix,P., Monperrus,M., and Baudry,B. (2021). Multi-Variant Execution at the Edge. *arXiv e-prints*, page arXiv:2108.08125.

[] Cabrera Arteaga,J., Floros,O., Vera Perez,O., Baudry,B., and Monperrus,M. (2021). Crow: code diversification for webassembly. In *MADWeb, NDSS 2021.*

[] Wen,E. and Weber,G. (2020). Wasmachine: Bring iot up to speed with a webassembly os. In *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 1–4. IEEE.

[] Chen,D. and W3C group (2020). WebAssembly documentation: Security. Accessed: 18 June 2020.

[] Cabrera Arteaga,J., Donde,S., Gu,J., Floros,O., Satabin,L., Baudry,B., and Monperrus,M. (2020). *Superoptimization of WebAssembly Bytecode*, page 36–40. Association for Computing Machinery, New York, NY, USA.

[] Cabrera Arteaga,J., Monperrus,M., and Baudry,B. (2019). Scalable comparison of javascript v8 bytecode traces. In *Proceedings of the 11th ACM SIGPLAN International Workshop on Virtual Machines and Intermediate Languages*, VMIL 2019, page 22–31, New York, NY, USA. Association for Computing Machinery.

[] Sengupta,S., Vadlamudi,S. G., Kambhampati,S., Doupé,A., Zhao,Z., Taguinod,M., and Ahn,G.-J. (2017). A game theoretic approach to strategy generation for moving target defense in web applications. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, page 178–186.

[] Okhravi,H., Rabe,M., Mayberry,T., Leonard,W., Hobson,T., Bigelow,D., and Streilein,W. (2013). Survey of cyber moving targets. *Massachusetts Inst of Technology Lexington Lincoln Lab, No. MIT/LL-TR-1166.*

[] Cox,M. R. (1893). *Cinderella: Three hundred and forty-five variants of Cinderella, Catskin, and Cap o'Rushes.* Number 31. Folk-lore Society.